Relational DB Design by ER- and EER-to-Relational Mapping

406.426 Design & Analysis of Database Systems

Jonghun Park

jonghun@snu.ac.kr

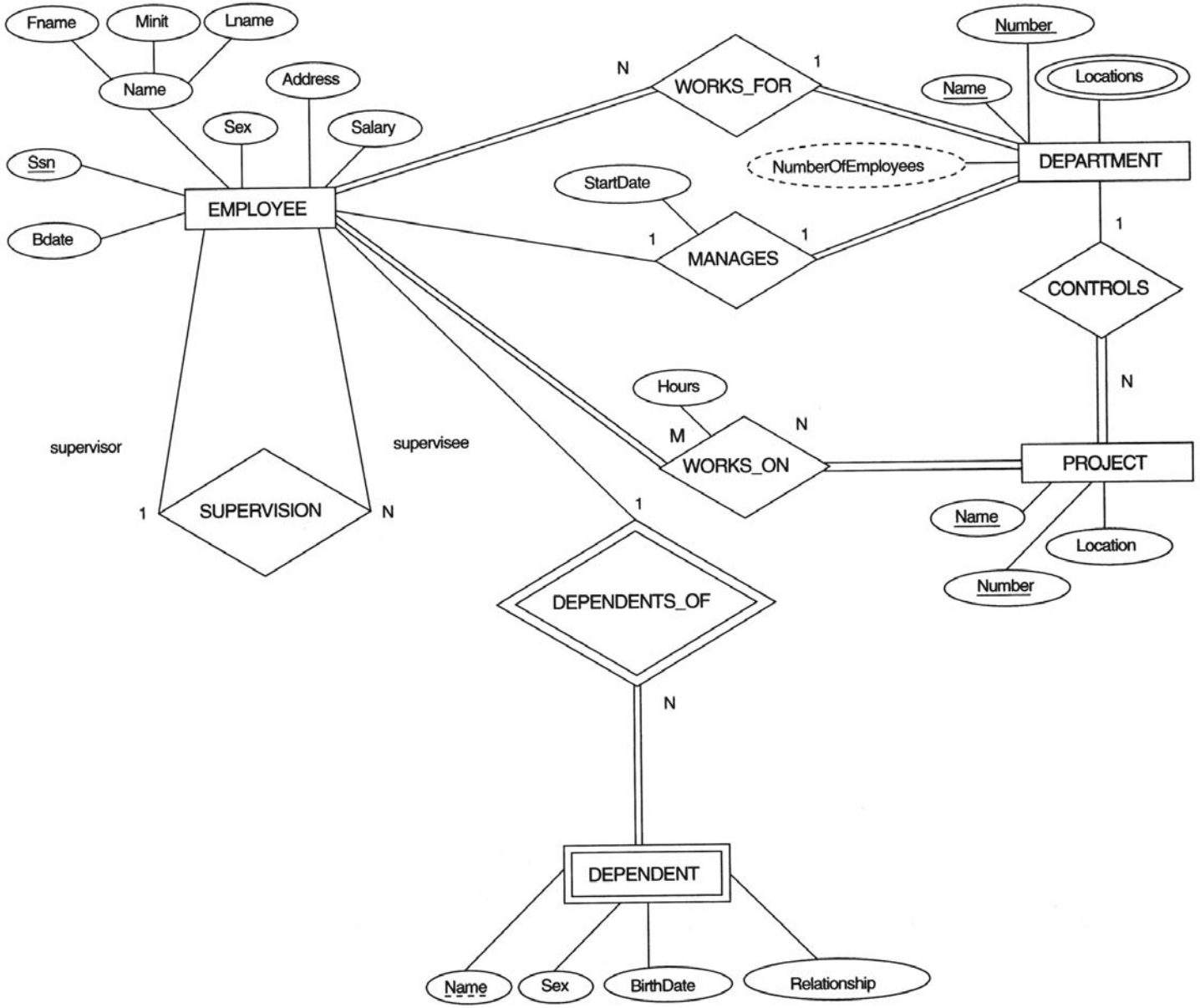
Dept. of Industrial Engineering

Seoul National University

outline

- ER-to-Relational mapping algorithm
  - step 1: mapping of regular entity types
  - step 2: napping of weak entity types
  - step 3: mapping of binary 1:1 relation types
  - step 4: mapping of binary 1:N relationship types
  - step 5: mapping of binary M:N relationship types
  - step 6: mapping of multivalued attributes
  - step 7: mapping of N-ary relationship types
- mapping EER model constructs to relations
  - step 8: options for mapping specialization or generalization
  - step 9: mapping of union types (categories)

example ER conceptual schema



the resulting relational database schema

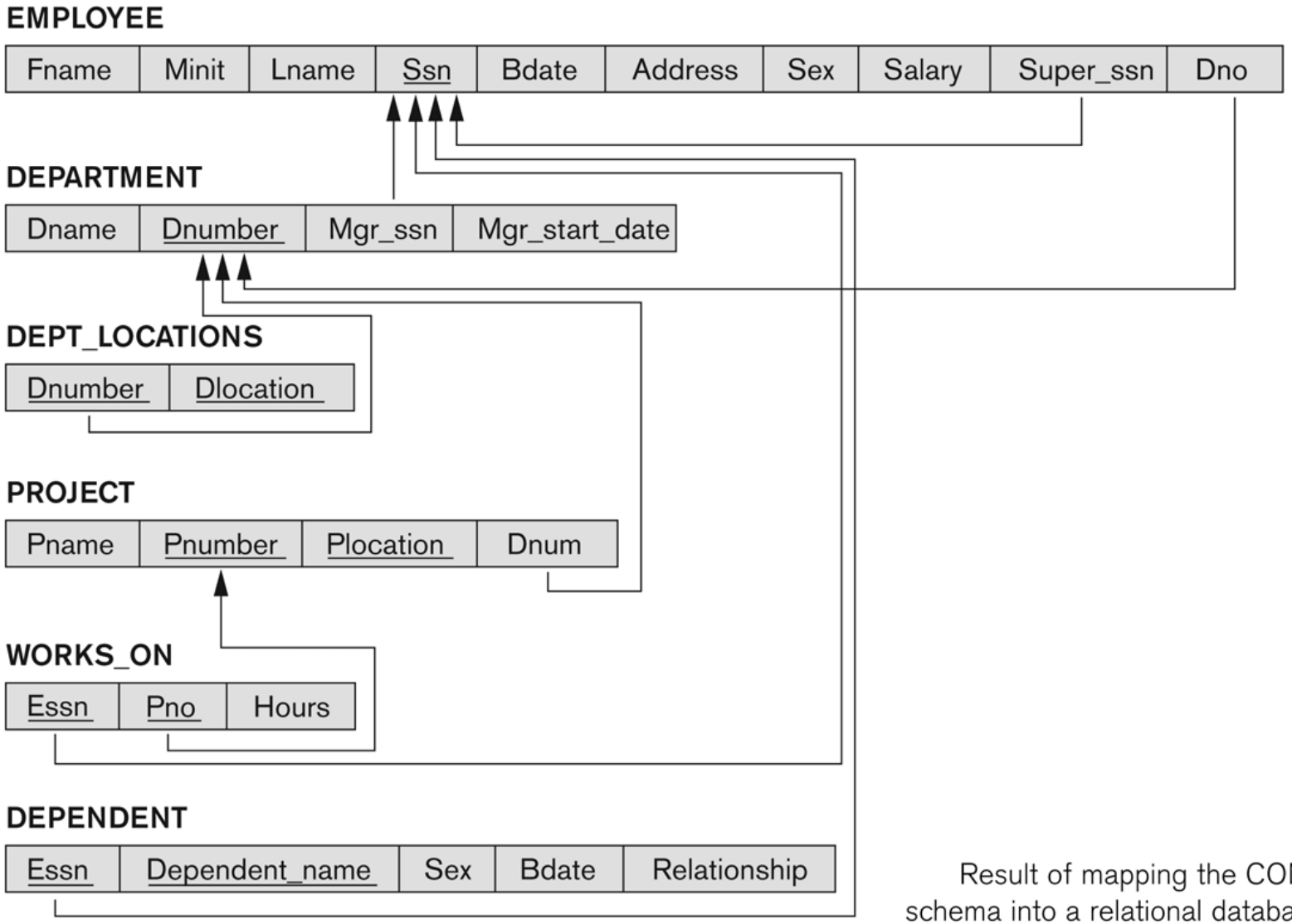
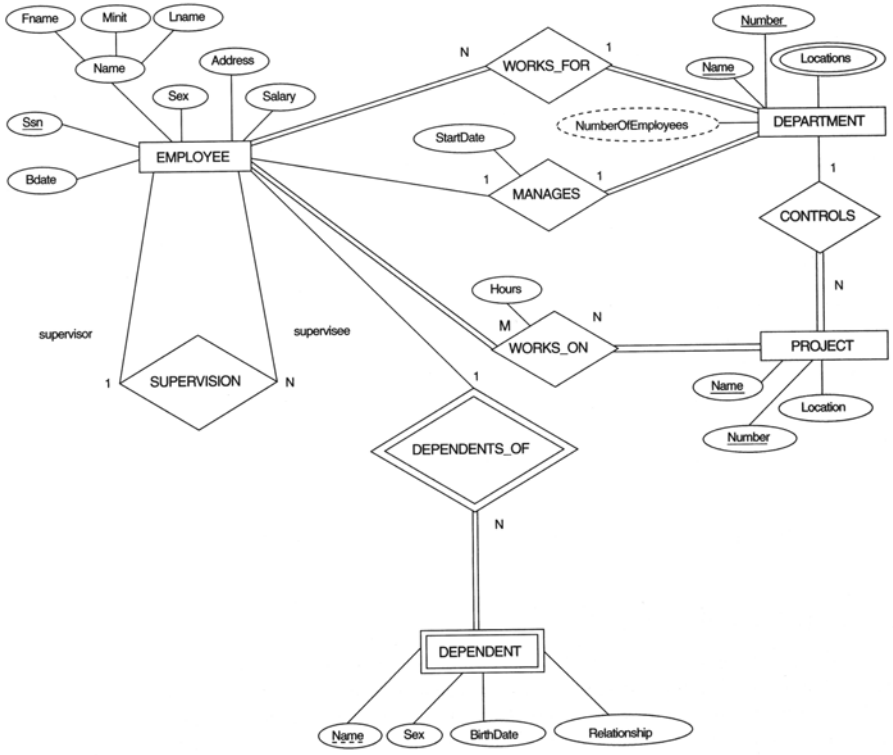
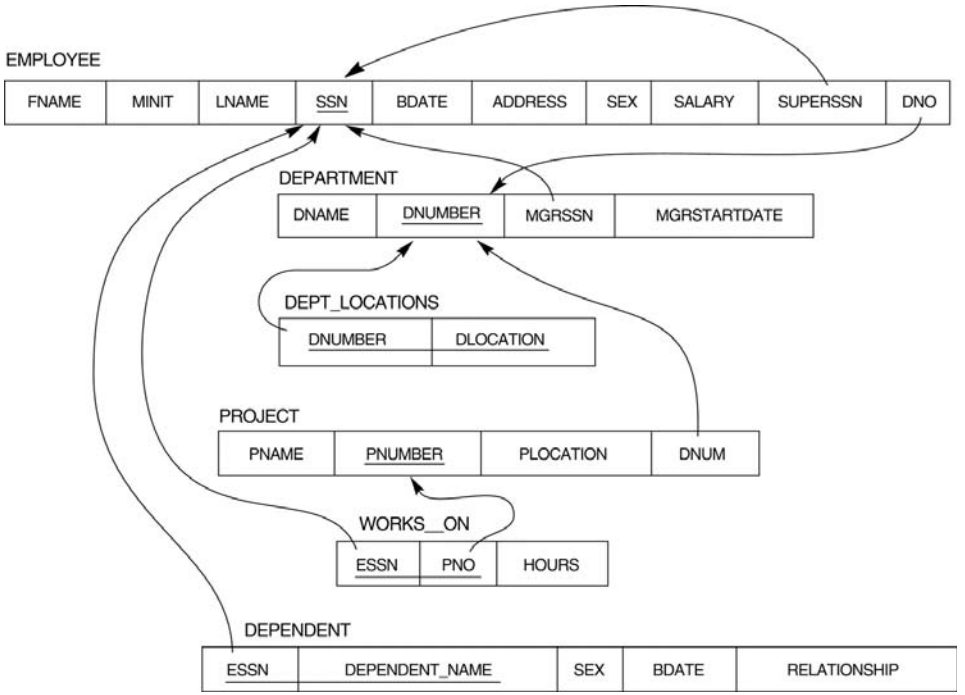


Figure 7.2
Result of mapping the COMPANY ER schema into a relational database schema.

together


Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
WORKS_FOR
N
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
MANAGES
1
1
1
CONTROLS
1
N
Hours
M
N
WORKS_ON
PROJECT
Name
Location
Number
supervisor
supervisee
1
SUPERVISION
N
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship


EMPLOYEE
FNAME
MINIT
LNAME
SSN
BDATE
ADDRESS
SEX
SALARY
SUPERSSN
DNO
DEPARTMENT
DNAME
DNUMBER
MGRSSN
MGRSTARTDATE
DEPT_LOCATIONS
DNUMBER
DLOCATION
PROJECT
PNAME
PNUMBER
PLOCATION
DNUM
WORKS_ON
ESSN
PNO
HOURS
DEPENDENT
ESSN
DEPENDENT_NAME
SEX
BDATE
RELATIONSHIP

an algorithm for ER-to-relational mapping

- step 1: mapping of **regular entity** types
  - for each regular entity type *E* in the ER schema, create a relation *R* that includes all the **simple attributes** of *E*
  - include only the **simple component attributes** of a composite attribute
  - choose one of the key attributes of *E* as **primary key** for *R*
  - if the chosen key of *E* is composite, the set of simple attributes that form it will together form the primary key of *R*
  - e.g., EMPLOYEE, DEPARTMENT, PROJECT

an algorithm for ER-to-relational mapping

- step 2: mapping of **weak entity** types
  - for each weak entity type *W* in the ER schema with **owner entity type** *E*, create a relation *R* and include all **simple attributes** of *W* as attributes of *R*
  - include as **foreign key attributes** of *R* the **primary key attribute(s) of the relation(s)** that correspond to the owner entity type(s)
  - the **primary key** of *R* is the **combination** of the primary key(s) of the owner(s) and the partial key of the weak entity type *W*, if any
  - if there is a weak entity type E_2 whose **owner is also a weak entity type** E_1, then E_1 should be mapped **before** E_2 **to determine its primary key first**
  - e.g., DEPENDENT

an algorithm for ER-to-relational mapping

- step 3: mapping of **binary 1:1 relationship** types
  - for each binary 1:1 relationship type *R* in the ER schema, identify the **relations *S* and *T*** that correspond to the entity types participating in *R*
  - **foreign key** approach
    - choose **one of the relations**, *S*, and include as a foreign key in *S* the primary key of *T*
    - include all the **simple attributes** of *R* as attributes of *S*
  - **merged relation** option
    - **merge** the two entity types and the relationship **into a single** relation
  - **relationship relation** option
    - set up a **third relation R** for the purpose of cross-referencing the primary keys of *S* and *T*
  - example: MANAGES -> DEPARTMENT.MGRSSN, DEPARTMENT.MGRSTARTDATE

an algorithm for ER-to-relational mapping

- step 4: mapping of **binary 1:N relationship** types
  - for each binary 1:N **relationship type *R***, identify the **relation *S*** that represents the participating entity type at the *N*-side of the relationship type
  - include as **foreign key** in *S* the **primary key of the relation *T*** that represents the other entity type participating in *R*
  - include any **simple attributes** of the 1:N relationship type as attributes of *S*
  - e.g., WORKS_FOR: S = EMPLOYEE, T = DEPARTMENT, DNO: the primary key of T

an algorithm for ER-to-relational mapping

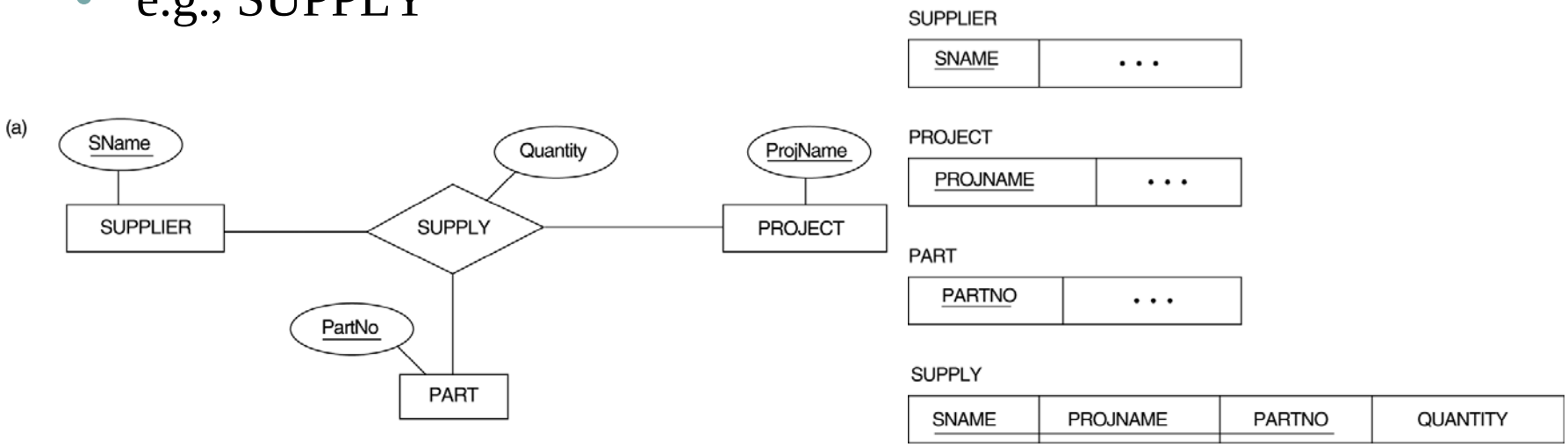
- step 5: mapping of **binary M:N relationship** types
  - for each binary M:N **relationship type *R***, create a **new relation *S*** to represent *R*
  - include as foreign key attributes in *S* **the primary keys of the relations** that represent the participating entity types
  - their combination will form the **primary key** of *S*
  - include any **simple attributes** of *R* as attributes of *S*
  - e.g., WORKS_ON: S = WORKS_ON

an algorithm for ER-to-relational mapping

- step 6: mapping of **multivalued attributes**
  - for each multivalued attribute *A*, create a **new relation *R***
  - *R* will include an attribute corresponding to *A*, **plus the primary key attribute *K* -** as a foreign key in *R* – of the relation that represents the entity type or relationship type that has ***A* as an attribute**
  - the primary key of *R* is the **combination of *A* and *K***
  - if the multivalued attribute is **composite**, include its simple components
  - e.g., Locations: *A* = DLOCATION, *R* = DEPT_LOCATIONS, *K* = DNUMBER

an algorithm for ER-to-relational mapping

- step 7: mapping of ***N*-ary relationship** types
  - for each *n*-ary relationship type *R*, where *n* > 2, create **a new relation *S*** to represent *R*
  - include as foreign key attributes in *S* the **primary keys of the relations** that represent the participating entity types
  - include any **simple attributes** of *R* as attributes of *S*
  - the primary key of *S* is usually a combination of all the foreign keys that reference the relations representing the participating entity types
  - e.g., SUPPLY



an algorithm for EER-to-relational mapping

- step 8: options for mapping **specialization** or **generalization**
  - convert each specialization with m subclasses $\{S_1, S_2, \ldots, S_m\}$ and superclass C, where the attributes of C are $\{k, a_1, \ldots, a_n\}$ and k is the key, into relation schemas using one of the following options:

an algorithm for EER-to-relational mapping

- option 8A: multiple relations-superclass and subclasses
  - **create a relation *L*** for C with attributes Attrs(L) = $\{k, a_1, \ldots, a_n\}$ and PK(L) = k
  - **create a relation L_i** for each subclass S_i, with the attributes Attrs(L_i) = $\{k\} \cup$ {attributes of S_i} and PK(L_i) = k
  - works for any specialization (total or partial, disjoint or overlapping)

EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

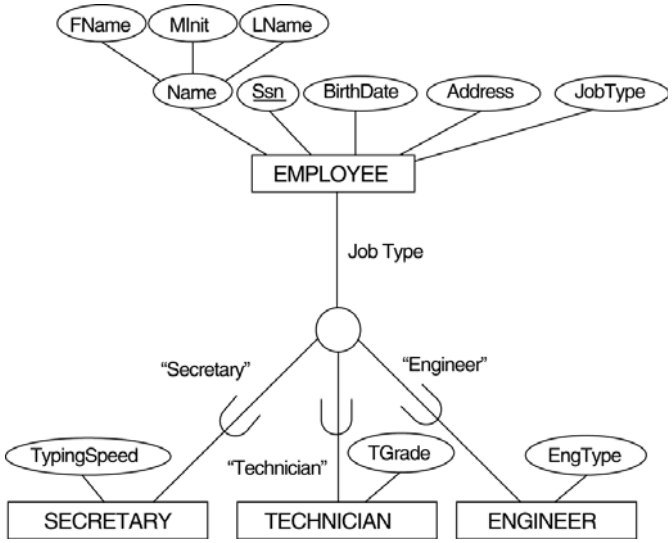
SSN	TypingSpeed

TECHNICIAN

SSN	TGrade

ENGINEER

SSN	EngType



an algorithm for EER-to-relational mapping

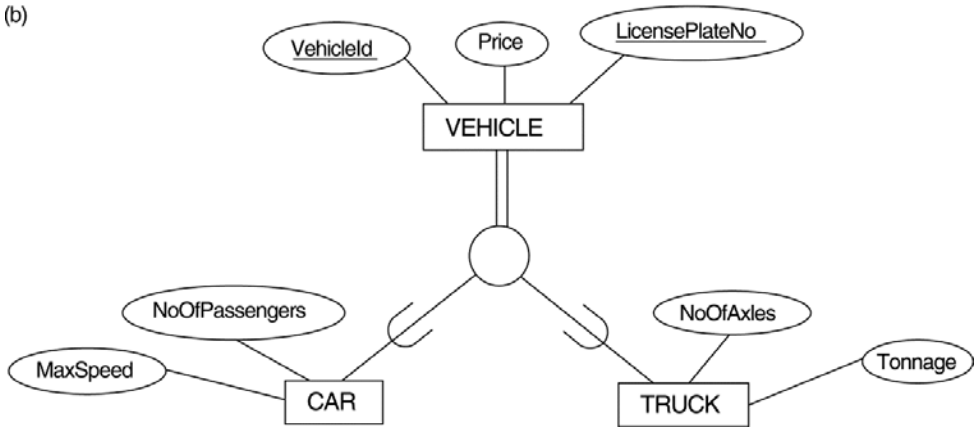
- option 8B: multiple relations-**subclass relations only**
  - **create a relation L_i** for each subclass S_i, with the attributes Attrs(L_i) = {attributes of S_i} $\cup$ {k, a_1, …, a_n} and PK(L_i) = k
  - only works for a specialization whose subclasses are **total**

CAR

VehicleId	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

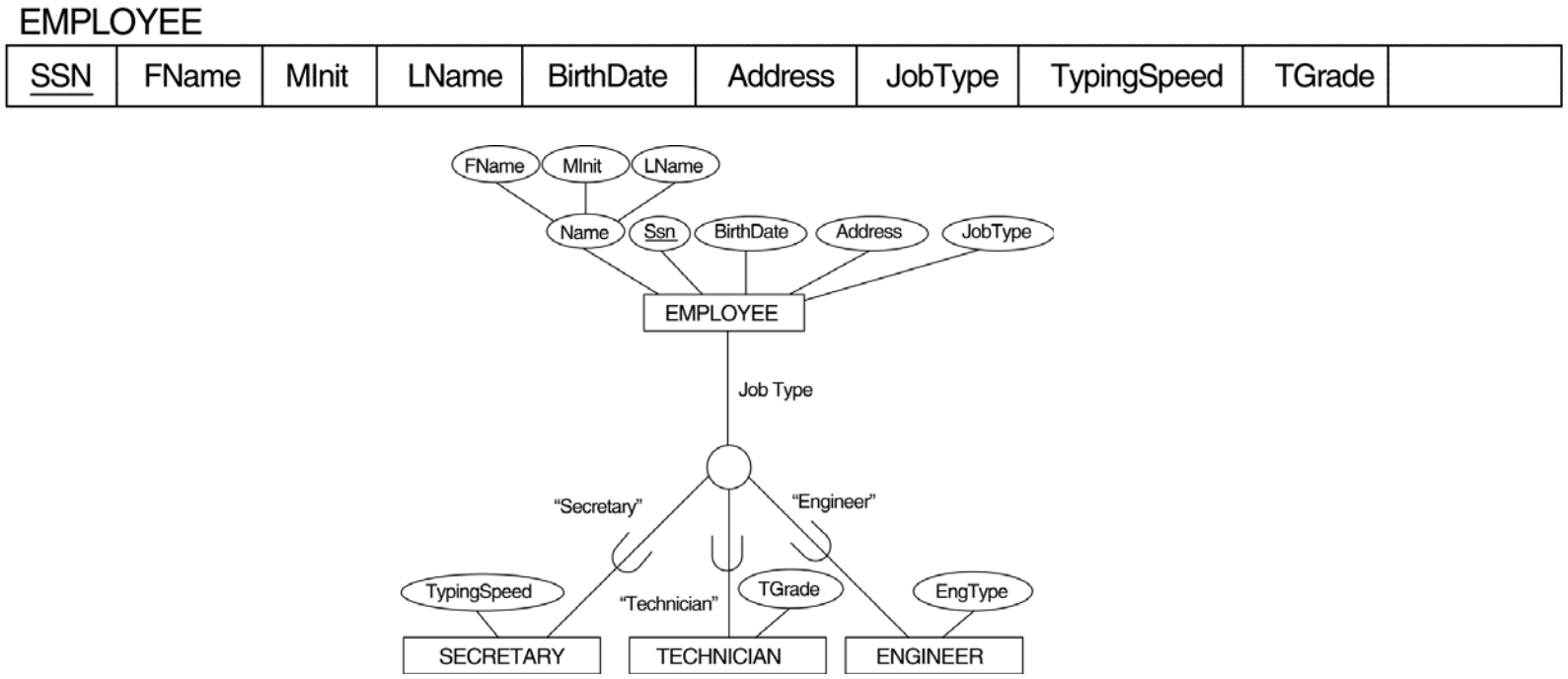
TRUCK

VehicleId	LicensePlateNo	Price	NoOfAxles	



an algorithm for EER-to-relational mapping

- option 8C: **single relation with one type attribute**
  - **create a single relation** L with attributes Attrs(L) = $\{k, a_1, \ldots, a_n\} \cup$ {attributes of S_1} $\cup \ldots \cup$ {attributes of S_m} $\cup$ **{*t*}** and PK(L) = k
  - the attribute t is called a **type attribute** that indicates the subclass to which each tuple belongs
  - works only for a specialization whose subclasses are **disjoint**

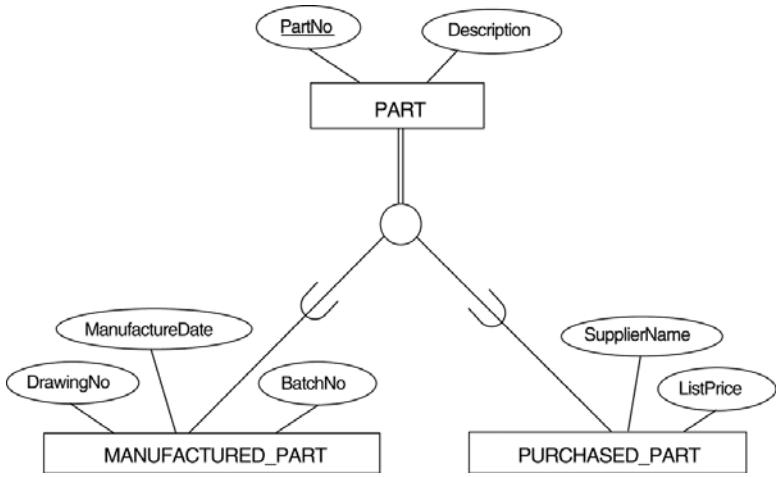


an algorithm for EER-to-relational mapping

- option 8D: s**ingle relation with multiple type attributes**
  - create a single relation schema L with attributes Attrs(L) = $\{k, a_1, \ldots, a_n\} \cup$ {attributes of S_1} $\cup \ldots \cup$ {attributes of S_m} $\cup$ $\boldsymbol{\{t_1, \ldots, t_m\}}$ and PK(L) = k
  - each t_i is a **Boolean type attribute** indicating whether a tuple belongs to subclass S_i
  - works for a specialization whose subclasses are **overlapping**

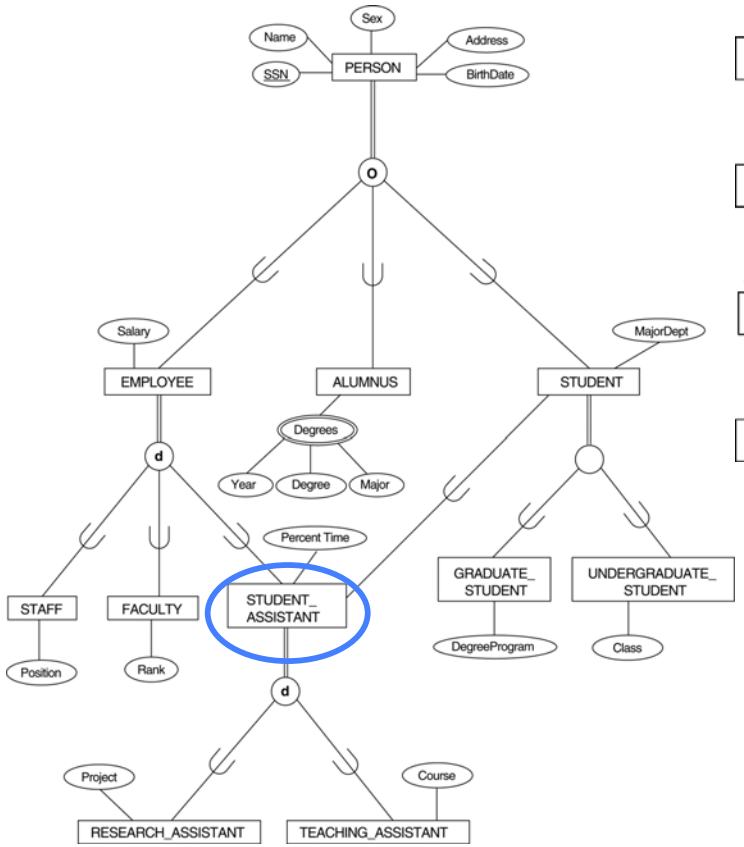
PART

PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice



mapping of shared subclasses

- **shared subclass**: a subclass of **several superclasses**, indicating multiple inheritance
- apply any of the options in step 8 to a shared subclass



PERSON

SSN	Name	BirthDate	Sex	Address

EMPLOYEE

SSN	Salary	EmployeeType	Position	Rank	PercentTime	RAFlag	TAFlag	Project	

ALUMNUS

SSN

ALUMNUS_DEGREES

SSN	Year	Degree	

STUDENT

SSN	MajorDept	GradFlag	UndergradFlag	DegreeProgram	Class	StudAssistFlag

mapping of categories

category?

a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types

- step 9: mapping of **categories**
  - mapping a category whose defining superclasses have **different keys**
    - specify a **new key attribute**, called a **surrogate key**
    - **include the surrogate key** attribute as foreign key in each relation corresponding to a **superclass** of the category
    - e.g., OWNER category
  - mapping a category whose **superclasses have the same key**
    - no need for a surrogate key
    - e.g., REGISTERED_VEHICLE

example for EER category mapping

PERSON

SSN	DriverLicenseNo	Name	Address	

BANK

BName	BAddress	OwnerId

COMPANY

CName	CAddress	OwnerId

OWNER

OwnerId

REGISTERED_VEHICLE

VehicleId	LicensePlateNumber

CAR

VehicleId	CStyle	CMake	CModel	

TRUCK

VehicleId	TMake	TModel	Tonnage	TYear

OWNS

OwnerId	VehicleId	PurchaseDate	LienOrRegular